Math 3040: Topics in Scientific Computing Introduction to finite element simulations using FEniCS and FreeFem++

M. M. Sussman

sussmanm@math.pitt.edu

Office Hours: 11:10AM-12:10PM, Thack 622

May 12, 2014

Topics

Introduction

Course topics
- An example
- The software

Virtual machines

Introduction to Unix

Who am I?

- Part-time faculty in Math Dept.
- Experience at Bettis lab
- Administer 2070/2071 Numerical Analysis lab
- Interested in numerical applications associated with fluid flow
- Interested in large-scale scientific computing

Prerequisites

- Some programming, knowledge of MATLAB preferred, C++ or Java helpful
- Calculus and linear algebra essential
- Some exposure to numerical analysis and computational math
- Some exposure to Linux
- No fear of the “command line”

Objectives

- Construct a computer FEM PDE model
- Combine knowledge of domain, math, computing
- Debugging computer models
- Leverage existing code
- Intro to parallel computing

References

1. FEniCS Book: Volume 84 of Springer Lecture Notes in Computational Science and Engineering series: Anders Logg, Kent-Andre Merdal, Garth Wells, “Automated Solution of Differential Equations by the Finite Element Method” ISBN: 978-3-642-23098-1 (Print) 978-3-642-23099-8 (Online)
   http://launchpad.net/fenics-book/trunk/final/+download/fenics-book-2011-10-27-final.pdf
2. FreeFem++ Book:
   http://www.freefem.org/ff++/ftp/freefem++doc.pdf
3. Reference: Hecht, F. New development in freefem++. J. Numer. Math. 20 (2012), no. 3-4, 251-265. 65Y15
4. **http://fenicsproject.org/qa/questions**
5. Recent Python and NumPy/SciPy books from **oreilly.com**
6. The Python Tutorial
   https://docs/python/org/2/tutorial
7. Tentative NumPy Tutorial
   http://wiki.scipy.org/Tentative_NumPy_Tutorial
8. Ubuntu forums **http://ubuntuforums.org/**
9. FEniCS list
10. FreeFem++ list

Format

- Monday-Thursday 12:30-2:15PM, Thackeray 524
- Office Hours: 11:10AM-12:10PM, Thack 622
- Available after class for questions
- Can't make office hours? Make an appointment

Grading

- Homework assigned from time to time
- Each homework problem is given a number of points
- Average on homework counts as 75% of grade
- Late homework accepted up to 1 week late
  - Penalty for late work is 20% of number of points
- Project counts as 25% of grade

Project

- List of possible project topics
  - Do you have a problem that fits with your research? Check with me for approval
- Can use either FEniCS or FreeFem++
- Project submitted as a “report”
  - Explain the task
  - Explain how problem is formulated
  - Include all input and supplementary code
  - Include output and plots
  - Explain the meaning of the results
  - Bibliography
  - Not graded on English usage
  - Should look like a thesis or published paper.

Topics

Introduction

Course topics
- An example
- The software

Virtual machines

Introduction to Unix

Course outline

1. Python programming language for scientific computing
2. Basic theory of the finite element method
3. FEniCS
4. FreeFem++

Topics

Introduction

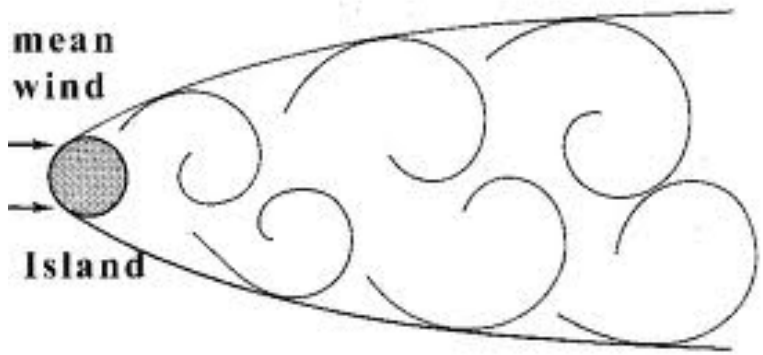
Course topics
- An example
- The software

Virtual machines

Introduction to Unix

von Kàrmàn vortex street

Fluid flow past an obstacle is not steady



Clouds

FreeFem++ live demo

`VortexStreet.edp`

Topics

Introduction

Course topics
- An example
- The software

Virtual machines

Introduction to Unix

What is the FEniCS project?

- **`http://fenicsproject.org/`**
- "The FEniCS Project is a collaborative project for the development of innovative concepts and tools for automated scientific computing, with a particular focus on automated solution of differential equations by finite element methods."

What is FEniCS itself?

- FEniCS core is a collection of software modules to formulate and solve a partial differential equation expressed in finite element form.
- Called from **Python** or C++
- Language for implementing weak form of PDEs
- Full power of Python for special-purpose tweaking
- Solve using open numerical software for linear algebra
- Reads files from mesh generators
- Writes files for plotters
- Runs on parallel computers
- 1D, 2D, 3D by changing 1 line of code

What is so great about Python?

- Widely used
- Freely available
- Not too difficult to learn
- General computer language
- Easily integrated with FEniCS modules

What is FreeFem++?

- FreeFem++ is a high level Integrated Development Environment for solving PDEs in 2D and 3D.
- Available for MS-Windows, Mac, Linux.
- Solve using UMFPACK and SuperLU
- Scripting language for weak forms is a dialect of C++
- Does its own mesh generation and plotting

C++

- Major programming languge
- Too difficult to present in a one-semester class
- Both FreeFem++ and FEniCS can be modified using C++
- Much easier to read than to write

Topics

Introduction

Course topics
- An example
- The software

Virtual machines

Introduction to Unix

How do I run these programs?

- Need consistent environment for this class
- Your own computer
  1. Install VirtualBox
  2. Copy VM or import package from me onto 32GB usb stick
  3. Insert usb stick
  4. Point VirtualBox at the VM on the usb stick
  5. Clone or import the VM onto your hard disk
  6. Boot the VM inside VirtualBox
- Computers on Thackeray seventh floor
  1. Copy VM from me onto 32GB usb stick
  2. Insert usb stick in computer in lab
  3. Point VirtualBox at the VM on the usb stick
  4. Boot the VM inside VirtualBox
  5. Set it up to copy files to *a different* usb stick

What is a Virtual Machine (VM)?

- VirtualBox runs on the host computer
- Pretends it is a whole computer inside a window
- VM has virtual display, disks, *etc.*
- VM can boot, shutdown, *etc.*
- Demo

What do I do next?

- Your VM is running a recent version of Kubuntu
- Log in as “student” (default)
- Password “math3040”
- KDE Linux desktop, “Start Menu” is at lower left
- You can add another user with a different name, if you wish

What things can I do?

- Browse the net
- Use Libre Office (MS Office capabilities)
- Install any new programs you like
- Run FreeFem++
  1. Open the Dolphin window manager (from favorites)
  2. Navigate freefem-examples → examples++-chapt3 → membrane.edp
  3. Double-click to bring up gui and run
- Run FEniCS
  1. Open the Dolphin window manager (from favorites)
  2. Navigate fenics-demos → documented → poisson → python → membrane.edp
  3. Double-click to bring up spyder and run

Command-line running

- Open a terminal window (from favorites) and run FreeFem++
  1. **`cd /freefem-examples`**
  2. Pick a group of examples *e.g.*, **`cd examples++-chapt3`**
  3. Pick an example: **`membrane.edp`**
  4. **`FreeFem++ membrane.edp`**
- Open a terminal window (from favorites) and run FEniCS
  1. **`cd /fenics-demos`**
  2. Choose a category: **`cd documented`**
  3. Choose a demo: **`cd poisson`**
  4. **`cd python`**
  5. **`python demo_poisson.py`**

Topics

Introduction

Course topics
- An example
- The software

Virtual machines

Introduction to Unix

Unix background I

- In 1969, Ken Thompson, Dennis Ritchie, Rudd Canaday, Doug McIlrow at Bell Labs split with the Multics timesharing project. They built an OS for the PDP-7 and Brian Kernighan named it Unix as a pun on Multics.
- The C programming language was developed as a generalized assembly language in order to write Unix.
- AT&T was constrained from marketing computer-related products because of an earlier consent decree, so they licensed the source code to universities for a pittance and to commercial entities. Thus, thousands of grad students added value.
- Berkeley (U. Calif.) became a developmental hotbed and ultimately developed the “Berkeley Systems Distribution” that contained many of the utilities now packaged with Unix (*e.g.,* **`vi`** and the C-shell). You may see the initials “bsd” crop up in many contexts.
- AT&T countered with “System V” (five). This contained the Bourne shell.

Unix background II

- Unix was designed as a multi-user, multi-tasking operating system from the ground up.
- The notions of “pipes” and “file redirection” are perhaps the most powerful concepts introduced in Unix.
- Linus Torvalds, as a grad student, took a course in OS
- He decided to write his own OS kernel
- Combined with the Gnu utilities, it is now called Linux

The File System: naming

- File names are of any length and case-sensitive. It is not a good idea to use spaces in a file name.
- Files often have an “extension” preceded with a dot, and these often are used to mean things. For example,

`.py`	Python source file
`.pyc`	compiled Python file
`.c`	C++ file
`.txt`	text file
`.gz`	compressed using **`gzip`**

- File extensions are conventions only. Text can just as easily be put into a file named **`report.o`** as **`report.txt`**.
- There can be multiple extensions.

Directories and navigation

- The forward slash `/` is used to separate directories.
- The current directory can be abbreviated as `.` and its parent can be abbreviated `..`
- An “absolute” directory path starts with a (forward) slash or a dot, a “relative” directory path does not.
- **`/home/mms125/math3040/test.py`**
- Directory changes are accomplished with the **`cd`** command (“Change Directory”).
- You create new directories with **`mkdir`** and remove them with **`rmdir`**.
- The default directory given to you when you log in is called your “home” directory. It is abbreviated either “**`$HOME`**” or “~”

Commands: format

- A “command” is a program that tells Unix to do something. It has the structure:

  command [options] [arguments]

- Options are generally preceded by a dash and can usually be strung together.
- Since Unix is multitasking, you can execute several commands at once. If you end a command line with an ampersand (`&`), the command will execute “in the background” and you will be able to continue typing other commands.
- You can bring a background job to the foreground with the command `fg`
- You can put a command currently in the foreground into the background by typing `C-z` to suspend it and then `bg`

Commands: file redirection

- Files can be “re-directed.” The “greater” symbol (>) between a command and a file name causes the command output to be written to a file. For example,

  ```
  ls > list.txt
  ```

  causes the list of your files to be written to a file named `list.txt`.
- This is a good way to capture the output from one of your programs.
- The “less” symbol (<) tells a command to take its input from a file rather than from the keyboard.
- The vertical bar (|) is called the “pipe” symbol. Placed between two commands, it causes the second to take its input from the output of the first.

Some common commands include:

`Ctrl-z`	suspend previous command
`bg`	continue suspended cmd “in background”
`fg`	put “background” command into “foreground”
`cd [directory]`	change directory
`cd`	change to home directory
`pwd`	print name of working (current) directory
`cp [options] file1 file2`	copy file1 to file2
`exit`, or `logout`	log out
`ls [options] [directory or file]`	list files in directory
`mkdir directory`	create directory
`mv [options] file1 file2`	move (rename) file1 to file2
`rm [options] file`	remove (delete) a file
`rmdir [directory]`	remove directory
`cat [file]`	copy a file to the screen
`less [file]`	display a file one page at a time
`!!`	re-do previous command
`!`*string*	re-do earlier command starting with *string*
`command > file`	output from command goes to file
`command < file`	input to command comes from file
`Ctrl-C`	abort current command